ESD-TR-84-326

Semiannual Technical Summary

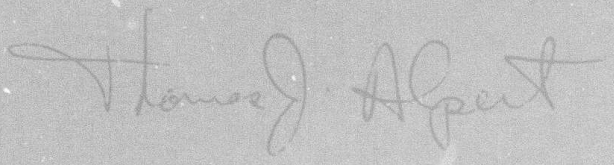
Distributed Sensor Networks

30 September 1984

Lincoln Laboratory
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LEXINGTON, MASSACHUSETTS

Prepared for the Defense Advanced Research Projects Agency
under Electronic Systems Division Contract F19628-85-C-0002.

The work reported in this document was performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology. This work was sponsored by the Defense Advanced Research Projects Agency under Air Force Contract F19628-85-C-0002 (ARPA Order 3345).



MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

DISTRIBUTED SENSOR NETWORKS

SEMIANNUAL TECHNICAL SUMMARY REPORT
TO THE
DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

1 APRIL — 30 SEPTEMBER 1984

ISSUED 29 MAY 1985

Approved for public release; distribution unlimited.

LEXINGTON MASSACHUSETTS

ABSTRACT

This report describes the work performed on the DARPA Distributed Sensor Networks Program at Lincoln Laboratory during the period 1 April through 30 September 1984.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

DISTRIBUTED SENSOR NETWORKS

I. INTRODUCTION AND SUMMARY

The Distributed Sensor Networks (DSN) program is aimed at developing and extending target surveillance and tracking technology in systems that employ multiple spatially distributed sensors and processing resources. Such a system would be made up of sensors, data bases, and processors distributed throughout an area and interconnected by an appropriate digital data communication system. The detection, tracking, and classification of low-flying aircraft have been selected to develop and evaluate DSN concepts in the light of a specific system problem. A DSN test-bed has been developed and is being used to test and demonstrate DSN techniques and technology. The overall concept calls for a mix of sensor types. The initial test-bed sensors are small arrays of microphones at each node augmented by TV sensors at some nodes. This Semiannual Technical Summary (SATS) reports results for the period 1 April through 30 September 1984.

Section II summarizes progress in the development of distributed tracking algorithms. Major software elements of a distributed tracking algorithm were completed and demonstrated to function correctly. Those portions that have been completed include modules for acoustic azimuth tracking, track initiation, internodal broadcast of azimuth measurements, association of measurements with tracks and for filtering azimuth measurements to update position tracks by means of Extended Kalman Filters. The algorithm components were tested and demonstrated to operate with real data as well as with simulated measurement data as input. Implementation of multi-site position combining and broadcast modules is in progress.

Several new display programs and a single computer UNIX-based test-bed simulation environment were implemented and used to aid in the development and testing of tracking algorithms. The simulation environment provides internodal and interprocess communications similar to that which will exist in the test-bed, and emulates the nodal system software environment by subroutine calls. The UNIX environment significantly aids the algorithm development process. It should be noted that the C-language code that implements the algorithm in the UNIX environment is the same code that will be used for test-bed nodes. The transition to the nodes will be primarily a recompilation and relinking task to create new load modules.

In addition, a paper was presented at the American Control Conference that describes the technical details of our distributed tracking algorithm.

Development and testing of distributed self-location algorithms is described in Section III. A distributed algorithm has been developed to process initial nodal position estimates and estimates of internodal ranges to obtain refined estimates of nodal positions.

Section IV summarizes results in the area of knowledge-based data interpretation. Emphasis has been upon the diagnosis or explanation of unexpected features in the tracking output from a

DSN system. Based upon our experience in developing an experimental rule-based diagnosis system, we concluded that the complex nature of a DSN system will make it very difficult to achieve good diagnostic performance with only empirical rules. We have therefore also formulated a design for a diagnosis system that incorporates model-based causal reasoning.

Progress with the development of a video sensor subsystem for the test-bed is summarized in Section V. The objectives are for the tracking system to cue the video subsystem, to improve position tracks by means of azimuth measurements extracted from video data, and to experiment with sensor resource management. Video sensor subsystem hardware has been purchased, assembled, and checked out by means of test software which we have written.

Section VI reviews progress with the acoustic subsystem for the test-bed nodes. Topics include the evaluation of wideband signal-processing algorithms, their real-time test-bed implementation, and minor changes in test-bed microphone hardware. Parametric sensitivities and azimuth estimation accuracy of the signal-processing algorithm were investigated. Azimuth accuracy ranged from a fraction of a degree to several degrees, depending upon target signal-to-noise ratio. Data from a subsonic but high-speed low-flying jet aircraft were part of the data base for these evaluations. This was the first application of the algorithm to these kinds of data and the results were completely satisfactory. A real-time version is now being developed for the test-bed nodes.

Test-bed facility enhancements are summarized in Section VII. This includes installation of a new VAX research computer, installation of an Ethernet that interconnects the VAX with other computational resources and with the test-bed nodes, addition of floating-point capabilities for nodal computers, and implementation of a broadcast protocol for radios that will be integrated into the test-bed.

II. DISTRIBUTED TARGET TRACKING

A test-bed simulator was implemented as a tool for debugging and evaluating the new tracking algorithms that will be installed in test-bed nodes. Nodal versions of most of the major elements of new distributed tracking algorithms were implemented and checked out using the simulator. Simulated acoustic data were used to test the portions of the new algorithms that have been implemented. New display software, compatible with the new tracking algorithm, was developed and used to review the output of the tracking algorithm. In June, a paper on the new tracking algorithm was presented at the American Control Conference.†

A. DISTRIBUTED TRACKING ALGORITHMS AND SOFTWARE

Figure II-1 shows the functional elements of tracking software in one node plus the major data flows within that node, between nodes, and with a User Interface Program (UIP). Interactions with the UIP are for control and display purposes and are not directly related to the tracking function. The existing UIP was modified during this report period to make it compatible with the new tracking software and display programs. Modifications consisted of decoupling the test-bed control and data spooling components of the UIP from communications functions and interfacing them with Ethernet and test-bed simulator software.

Each functional element illustrated in Figure II-1 was implemented using two or more communicating processes. In each case, one of those processes serves as an "impedance matcher" for communications with the UIP. The tracking software communicates internally via binary-encoded messages, but the UIP communicates via ASCII-encoded messages. The impedance matcher performs the necessary conversions.

All of the elements in Figure II-1 are complete except for the tracking algorithm which is not in a final state but can perform tracking functions. The following describes the tracking algorithm and its status in more detail.

Figure II-2 illustrates the components of the tracking algorithm. Two data bases are maintained in each node: one for acoustic azimuth tracks and one for target position tracks. Acoustic azimuth tracks include estimates of azimuths and azimuth rates. They also include a covariance estimate for both azimuth and azimuth rate. Acoustic azimuth tracks are formed by applying a Kalman Filter to azimuth measurements. They are broadcast if they satisfy accuracy and elapsed-time criteria since the last broadcast.

Local azimuth tracks are combined with azimuth tracks received from other nodes to initiate position tracks. The position tracks estimate target positions and velocities, and also include covariances of those estimates. By applying an Extended Kalman Filter to azimuth measurements,

---

† R.R. Tenney and J.R. Delaney, "A Distributed Aeroacoustic Tracking Algorithm," Proceedings of the 1984 American Control Conference, San Diego, California, 6-8 June 1984, Vol. 3, pp. 1440-1450.

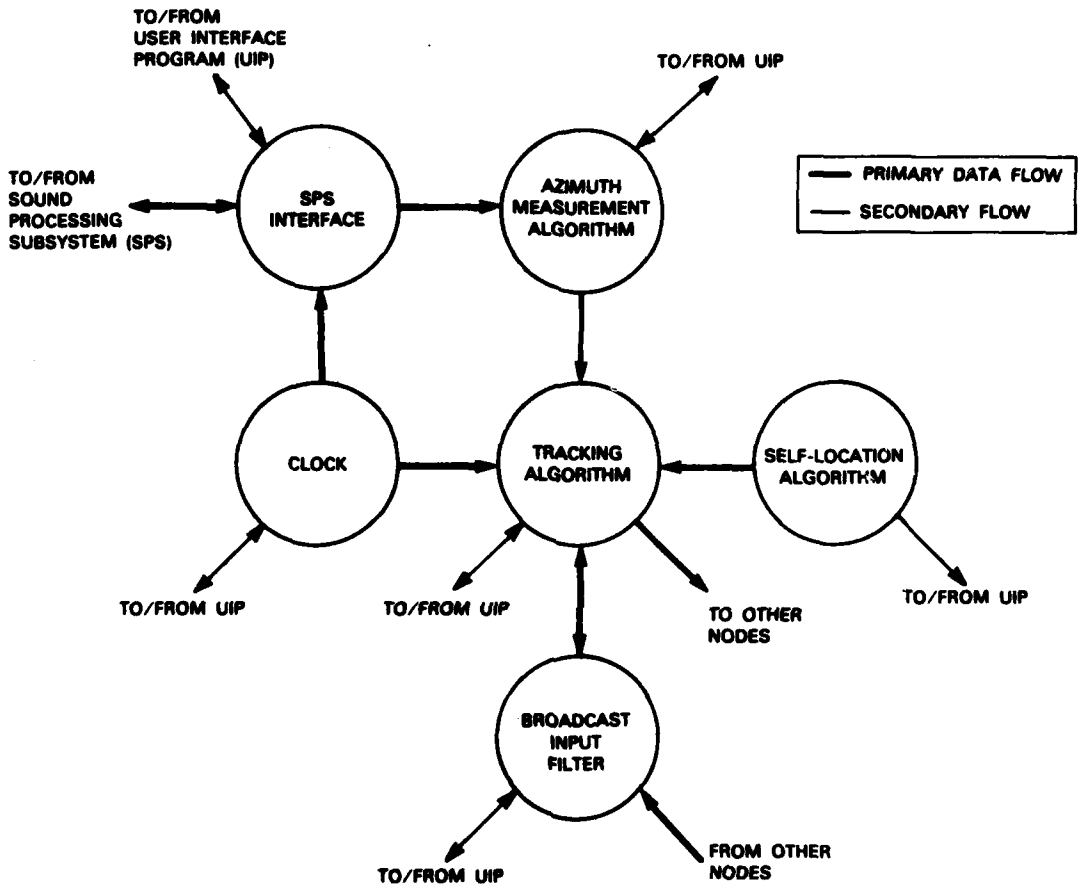
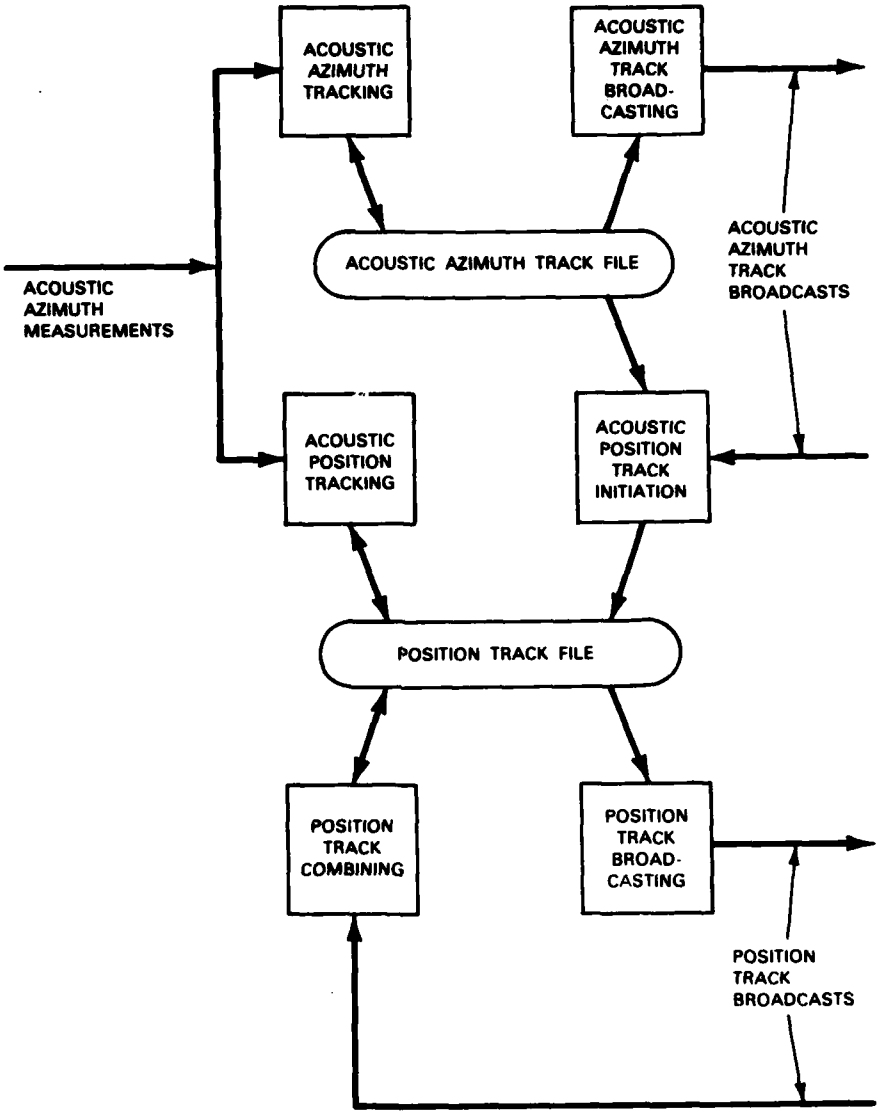


Figure II-1. Functional organization and data flows for new tracking software.



149934-N

Figure II-2. Components of new tracking algorithm.

position tracks are updated. They are also updated using position tracks received from other nodes. Position tracks are broadcast if they satisfy accuracy, elapsed time since last broadcast, and other criteria. These aspects of the new tracking algorithm were described in detail in a previous SATS.†

The tracking algorithm implementation is complete except for position track combining and broadcast modules. These modules will be relatively small but will add significantly to the complexity of system behavior. Therefore, the tracking algorithm has been carefully tested and studied without these components to minimize the difficulty of integrating them into the system. In the course of that study, design and implementation changes were made which reduced the number of false tracks and improved the quality of the true tracks. Implementation of the two missing components began late in this reporting period.

B. TRACKING ALGORITHM PERFORMANCE

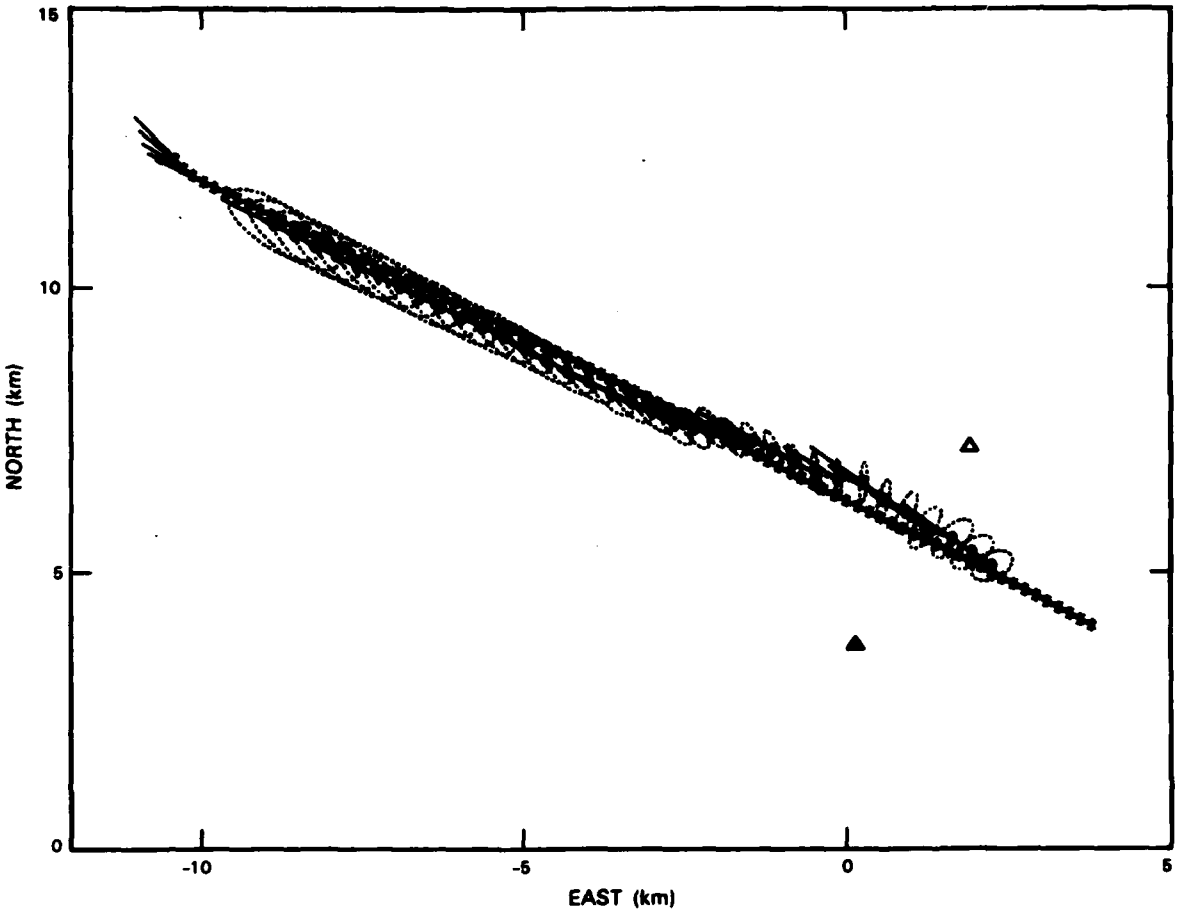
The tracking algorithm creates and maintains useful position tracks despite the absence of the position track combining and broadcasting modules. The exchange of azimuth tracks allows position tracks to be initiated, and the acoustic position tracking algorithm updates those tracks using local azimuth measurements. The resulting tracks are not as accurate as those which would be obtained if position tracks were also exchanged. But they can be quite accurate, as Figure II-3 illustrates. The results are for a jet aircraft flying at Mach 0.6 at low altitude. They were obtained using data collected by two acoustic arrays separated by about 4 km.

The asterisks in Figure II-3 indicate the path actually flown by the aircraft. The first asterisk indicates the position at the time the data acquisition system was turned on. Roughly 2 s of travel separate each asterisk. The nodes are indicated by triangles, with a "+" symbol in the middle of the one from which the azimuth measurements are taken for acoustic position tracking. Position estimates are indicated by small circles, each with a "tail" indicating estimated heading and speed. A dotted ellipse surrounds each circle; it is the two-dimensional equivalent of one standard deviation "error bar" about the estimated position. Again, roughly 2 s separate each estimate. A dotted line used to connect estimates is often obscured by the tails.

Several measurements were required to establish azimuth tracks accurate enough to broadcast and to form the basis for an initial position estimate. This explains the delay in track initiation, although the target was detected by both arrays from the moment the data acquisition system was turned on. For these data the initial position errors of a few hundred meters are first reduced, and then slowly grow as the target moves far beyond the nodes. The average error is roughly 200 m, even when the aircraft's range is as much as 10 km.

The tracking software was also exercised on real and simulated azimuth measurement data for lower velocity targets such as helicopters, including one case with two simulated targets. The

† Semiannual Technical Summary, Distributed Sensor Networks Program, Lincoln Laboratory, M.I.T. (30 September 1983), DTIC AD-A146209/2.



149935-N

Figure II-3. Position track of Mach 0.6 jet aircraft.

track accuracy was generally worse than for the high-speed jet, the root-mean-square error being typically 500 m. This result was expected for these relatively slow-moving targets without exchanging position tracks. Theory predicts that, without position track combining, lower-velocity targets are less observable than high-speed targets.

C. DISPLAY SOFTWARE

In addition to illustrating the tracking software performance as of 30 September, Figure II-3 also illustrates the kind of displays generated by one of four new display programs that have been implemented. The plot shown was generated in real-time sequence, with actual target positions (asterisks) and the estimates (circles, etc.) appearing alternatively in proper time order. The quality of a track can be judged by the position of the asterisks inside (or outside) the ellipses. Actual positions are plotted only when provided from an external source. For Figure II-3, such information was obtained from independent radar tracking. Information also is provided by data simulation software when simulated data are used as input to the tracker.

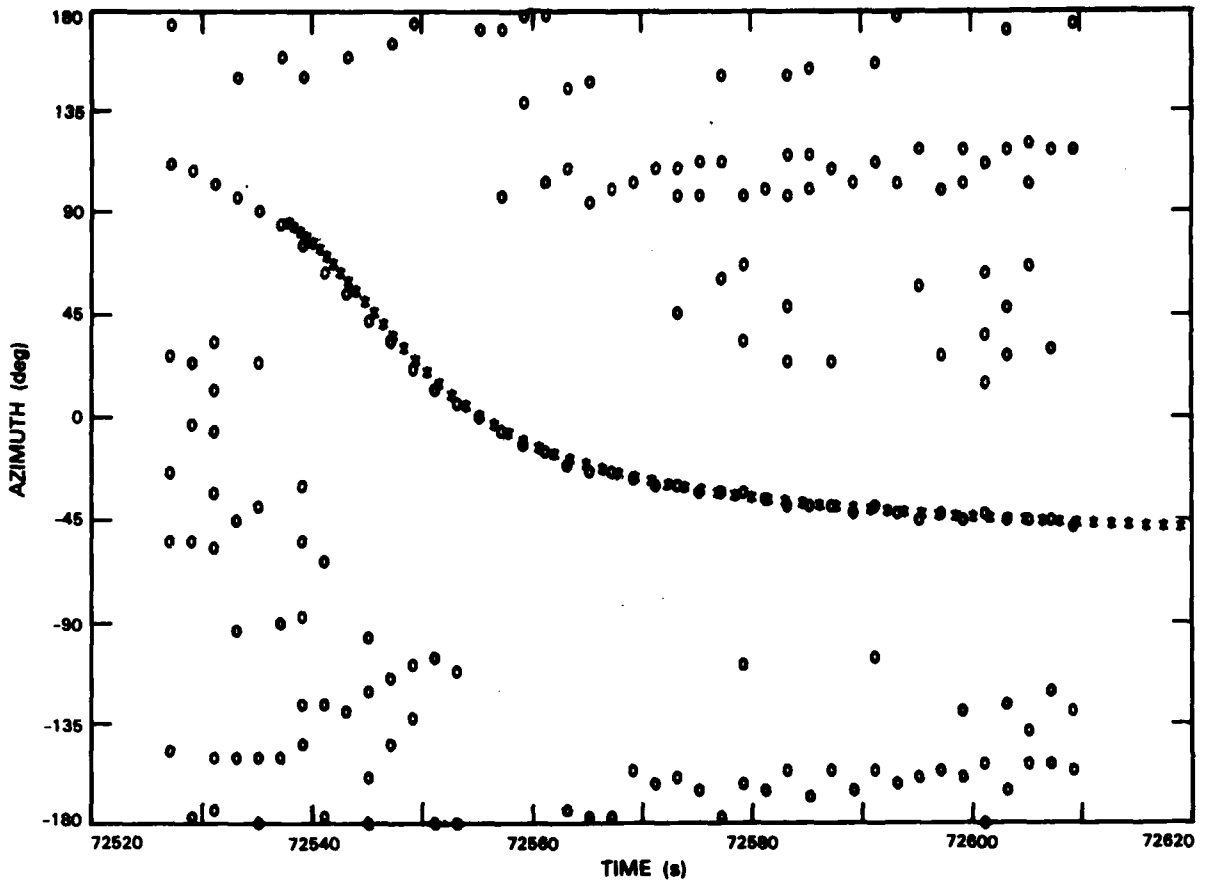


Figure II-4. Signal processor output display for jet aircraft.

Three other kinds of plots are made by the display programs in the same real-time manner (see Figures II-4, II-5, and II-6). They correspond to Figure II-3 in that they plot results for the same experiment and the same node.

Figure II-4 plots the output of the signal-processing algorithm. This is the data flowing between the Sound Processing Subsystem (SPS) interface and the azimuth measurement algorithm in Figure II-1. The vertical axis is the azimuth measured by the algorithm, and the horizontal axis represents time. The asterisks indicate the measurement predictions based upon the actual target track. The circles represent the detections made by the signal-processing algorithm.

As many as eight detections are produced every 2 s. At most, one detection is caused by the target at each time; the others are noise. The time misalignment of the first asterisk and the first circle is caused by propagation delay. The sound received at 72528 s was emitted by the aircraft more than 10 s earlier. By the time the sound emitted by the target at that time reaches this node, the target has moved some 2 km down its flight path. Detections cease at 72608 s because only 80 s of data were processed.

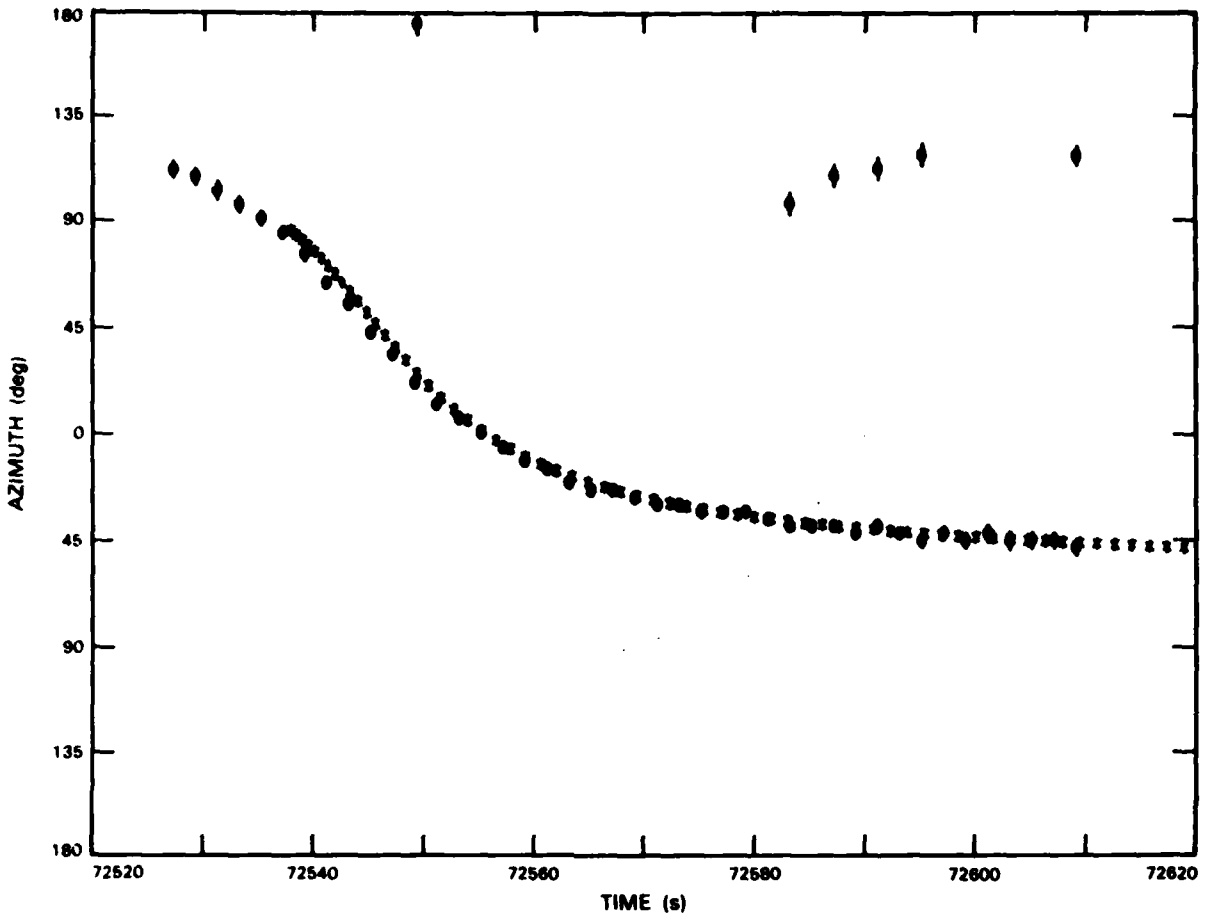


Figure II-5. Azimuth measurements display for jet aircraft.

Figure II-5 illustrates the data flowing from the azimuth measurement algorithm to the tracking algorithm. The data stream is a conditioned version of the raw measurement stream. The azimuth measurement algorithm discards raw measurements which seem unlikely to have been caused by aircraft and associates a variance with each measured azimuth. The format of the plot is similar to that in Figure II-4, but it appears different because most of the false detections have been discarded and because a vertical line segment has been added to each circle. These vertical line segments represent one standard deviation error bars. Note that the azimuth measurement algorithm is not perfect; several false azimuth measurements are passed to the tracking algorithm.

Figure II-6 shows the azimuth tracks. Symbology has been added beyond that which appears in Figure II-5. Tails have been added to the data points. The slope of each tail indicates the azimuth rate. In addition, dotted lines connect sequences which form one track. Note that the false azimuth measurements shown in Figure II-5 lead to the two false azimuth tracks in this figure.

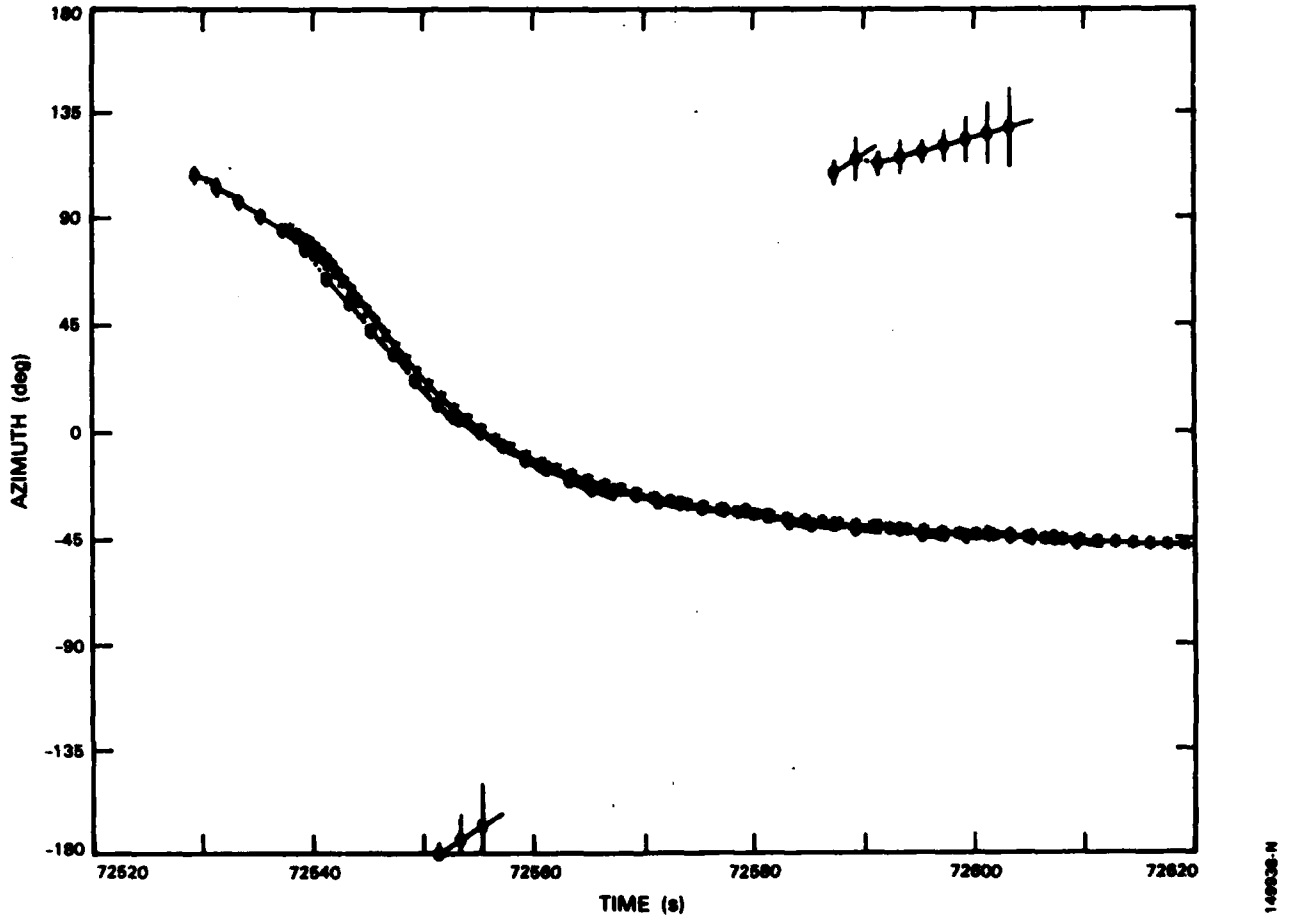


Figure II-6. Azimuth tracks display for jet aircraft.

D. TEST-BED SIMULATION

Figure II-7 depicts the test-bed software configuration with an emphasis on tracking software. Each node consists of a SPS that provides target detections and a Standard Nodal Computer (SNC) that performs tracking. Within each SNC, the operating system provides a two-layer interface to an Ethernet that links the nodes with each other and with the User Interface Computer (UIC). One layer adds and removes communication headers from application-level messages while the other transmits and receives the messages. The UIC runs the UIP, which controls the tracking software and records its performance, and the display programs, which selectively show tracking algorithm performance. The Ethernet interface in the VAX sends and receives messages. Header addition and removal is done in the UIP.

The test-bed provides limited support for software debugging. This has prompted us to develop a simulation of the test-bed having sufficient fidelity to allow most of the debugging of the tracking software to be done on a general-purpose research computer. This simulation was initially developed under Version 7 UNIX on a PDP-11/70 and is now being converted to run under Berkeley 4.2 UNIX on a PDP-11/780. In addition to aiding in debugging, the simulation provides an environment in which to exercise the tracking software in advance of the actual test-bed's availability.

Figure II-8 illustrates the test-bed simulation configuration. The most obvious component is the Ethernet and communications interface simulation, indicated by the odd-shaped box in the figure. This component mimics the test-bed Ethernet distribution of point-to-point and broadcast messages, plus the addition and removal of communication headers by the Nodal Run-Time System (NRTS) in each nodal SNC.

Another important element of the simulation environment is the acoustic measurement interface. This mimics the acoustic measurement process by routing preprocessed or simulated signal processor output from a file into the appropriate port of the SPS interface.

In the simulation environment, NRTS operating system services are emulated by subroutines with identical parameters to the NRTS calls and by following special conventions in writing the tracking software. Differences between the simulation and test-bed node environment are isolated in a small number of subroutines. Because of such residual differences, some debugging must be done in the test-bed. But the tracking algorithm itself is being completely debugged in the simulation environment.

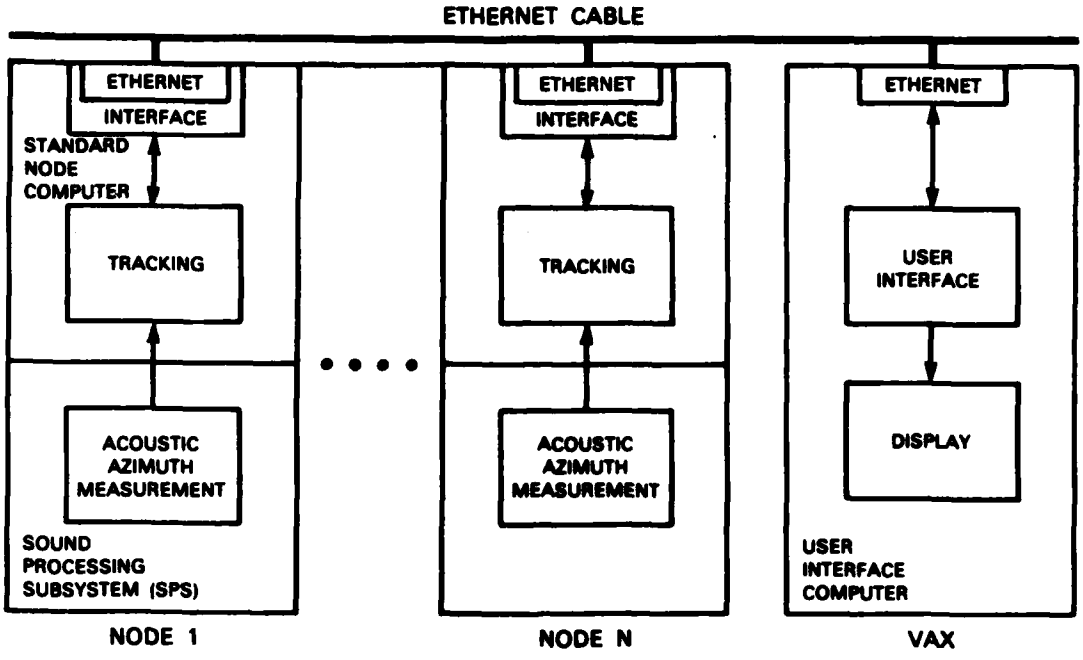


Figure II-7. Test-bed software organization.

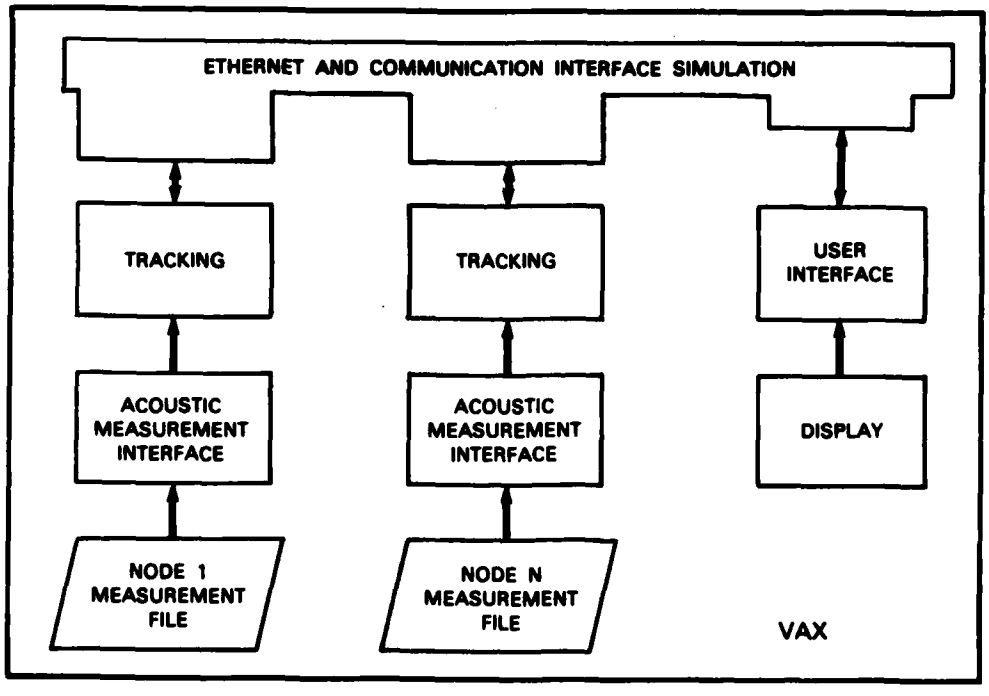


Figure II-8. Test-bed simulator software organization.

III. SELF-LOCATION

The DSN self-location problem has been formulated as a distributed estimation problem. Two algorithms based upon estimation theory have been implemented and are being tested. The algorithms require initial nodal position estimates, and the network must have sufficient connectivity.

The two algorithms correspond to two different ways of finding the distance between two nodes, as shown in Figure III-1. The first uses the differences in x and y positions of two neighboring nodes as the measurements. These measurements are a linear function of nodal positions. The second uses the actual range measurements, which can be formed using time of-arrival data. This makes the estimation problem nonlinear in the x and y positions. The algorithms solve the two-dimensional distributed location problem. In both algorithms, it was assumed that all measurements would have errors, and a zero-mean Gaussian error distribution was used.

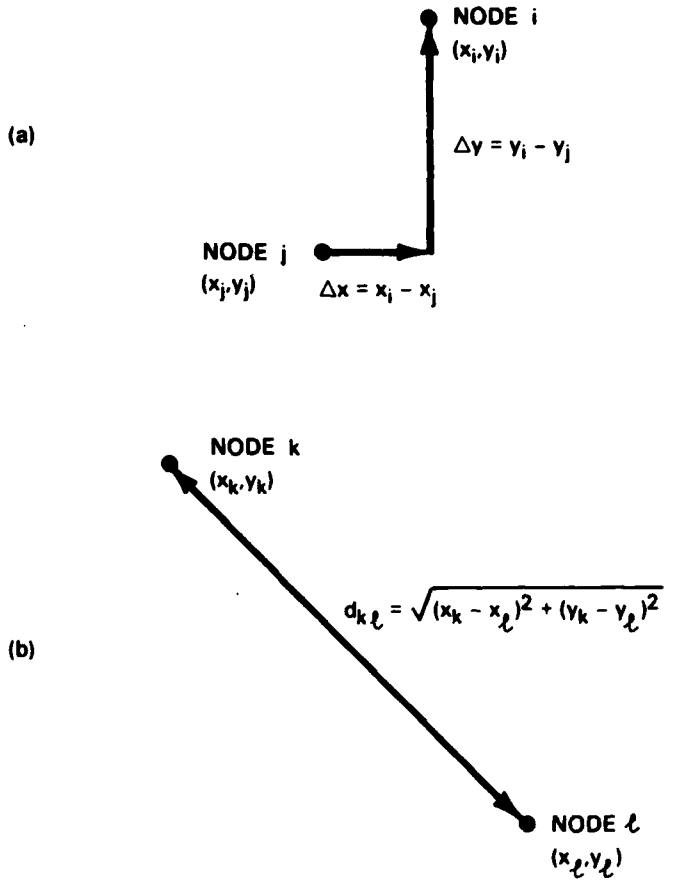


Figure III-1. Two types of internodal measurements: (a) linear; (b) nonlinear.

TABLE III-1

Algorithm Performance for Network in Figure III-2

Measurement Standard Deviation (m)	Initial Position Standard Deviation		
	5 m	50 m	500 m
150	3[1] 6.4[2]	4.6 57	25.2 300
15	4 5.8	17.6 35	37.2 48
1.5	11.4 3.5	29.2 5.3	43.8 4.4

Notes: (1) Average number of algorithm cycles per node.
(2) Final position error standard deviation (in meters).

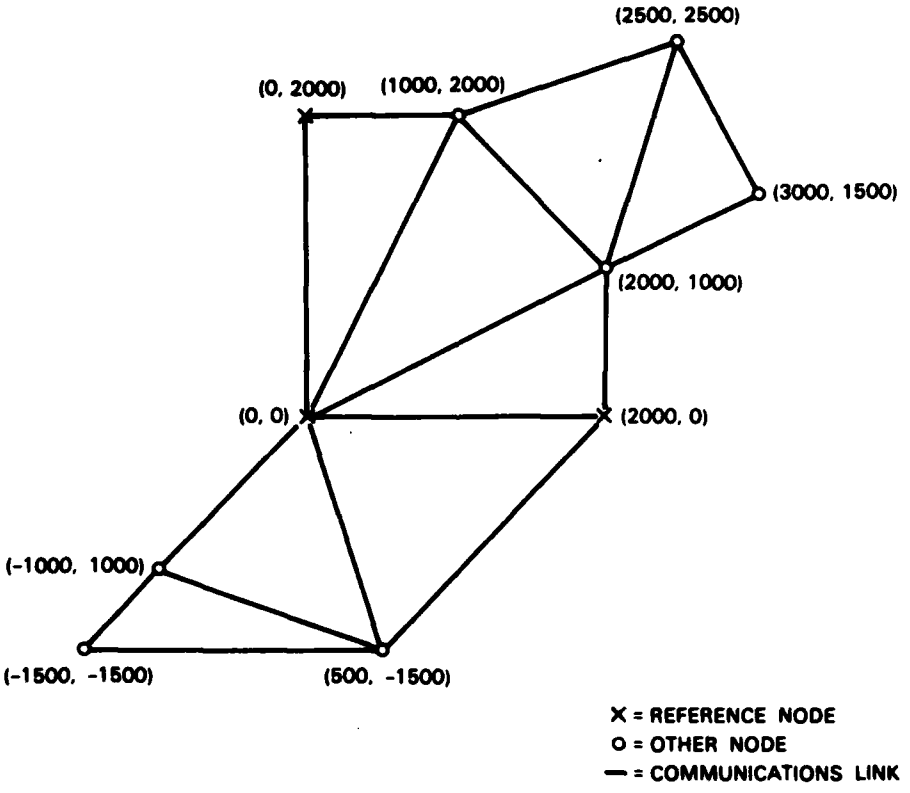


Figure III-2. Sparsely connected network.

The linear problem was treated first because it is simpler to solve, even though that kind of measurement would not be available in a real network. Tests of this algorithm confirmed the validity of our approach and allowed us to develop a test environment for distributed self-location algorithms that use range measurements. The variables in the simulation were: the standard deviation of node initial position estimates, standard deviation of internodal measurements, and a parameter controlling convergence of the algorithm at each node. Random 10-node networks were generated with 1 node considered as a "reference" node (with a smaller standard deviation on its initial estimate than the other nodes) to prevent the network configuration from "drifting" in absolute coordinates. Ten runs were made on each of ten networks for three different values of both prior and measurement error standard deviations. All cases behaved much as one would expect intuitively, taking more time to converge when connectivity was low or when errors were large.

The more complex nonlinear algorithm that makes use of range measurements is based upon the minimization of a fourth-order cost function of the node positions. The derivatives of this function are set to zero to find the minimum. As a result, there are two simultaneous cubic equations to be solved at each node at every iteration. Three different solution methods were investigated. The most successful was to solve for each position variable separately with the other held constant at its last value, then iterate between the x and y unknowns. Newton's method was less successful because of its tendency to find local instead of global extreme points. The third method was to find a closed-form solution with the help of MACSYMA (a symbolic mathematics manipulation program), but this was unsuccessful due to the extreme complexity of the equations that were obtained.

Similar simulation testing was done for the nonlinear algorithm with the exception that three reference nodes were needed to prevent rotation and translation of the solution. Again, the algorithm performed as expected. Table III-1 shows some of the performance measures found for the network in Figure III-2. Note that none of the nodes can hear all of the reference nodes, so no node is performing triangulation from three known positions. The results shown required about 6 s of VAX-11/780 time per algorithm cycle per node.

IV. KNOWLEDGE-BASED DATA INTERPRETATION

During this report period, we have investigated the expectation-and-diagnosis approach to knowledge-based data interpretation. In particular, we have completed experiments with rule-based diagnosis and have formulated a detailed design for a diagnosis system that uses model-based causal reasoning. We have also identified options for providing expectations that can be used for diagnosis.

A. RULE-BASED DIAGNOSIS

Rule-based systems are the most developed off-the-shelf AI technology. During this period we obtained YAPS, which is a general package suitable for implementing antecedent-driven rule systems, and used it to build a DSN diagnosis system consisting of approximately 200 rules. Standard knowledge engineering techniques were used to obtain the empirical rules-of-experience. The diagnosis system was implemented on our Symbolics 3600 Lisp machine.

A DSN system involves many complex interacting phenomena. It is extremely difficult to represent the knowledge needed to diagnose such a system in simple rules of the type allowed by generic expert system tools such as YAPS. This became quite apparent as we developed the DSN rule-based diagnosis system. One aspect of the problem is that empirical rules-of-experience may no longer be valid for even small changes in the details of the system. This is because empirical rules take a black-box view of the system and concentrate on input-output relationships. If the internal design of the DSN system is changed, it is then not clear how to change these input-output relationships. It is more preferable to have a diagnosis system which has model-based knowledge of how the DSN functions.

B. DSN DIAGNOSIS WITH MODEL-BASED KNOWLEDGE

Each node of the DSN system contains complex parameterized algorithms for signal processing and tracking. Whenever these parameter settings are not suited to the actual situation, errors arise in the system outputs. In diagnosing such a system, one can often make use of theoretical models underlying the algorithms. For example, underlying the signal processing is the theory of spectral and wavenumber analysis and the DSN tracker makes use of Kalman filtering theory. But note that whereas the underlying theories supply conceptual models, they do not spell out how to use those models for diagnosis. We have addressed this issue and have designed a diagnosis system with knowledge structures that will allow us to use the conceptual models to accomplish DSN diagnosis.

A strategy known as causal backward tracing uses model-based knowledge and has previously been investigated for diagnosing complex systems. Lesser and Hudlicka† used this strategy with the assumption that all intermediate data states are always available. This is unrealistic

† E. Hudlicka and V. Lesser, "Meta-Level Control Through Fault Detection and Diagnosis," in *Proceedings of AAAI-84* (August 1984), pp. 153-161.

for our DSN system because of the enormous amounts of data and, in the absence of the data, causal backward tracing often cannot be accomplished. For our work we have limited the available data to consist of only the target position tracks generated by each DSN node.

For situations where intermediate data states are not available for inspection, Davis† (in the context of digital hardware diagnosis) used a simulation strategy for generating the missing data states. There are difficulties with this approach for our DSN diagnosis problem. First, it might be necessary to simulate an enormous amount of data. Second, and more important, actual DSN data often do not efficiently capture the causality of events as explained by the conceptual models.

Our approach is based on conceptual simulation instead of the direct simulation approach of Davis. The focus is upon data structures that are appropriate to the underlying conceptual models. We view the DSN as an operator that maps actual aircraft tracks into output tracks. Ideally, it is an identity operator. Diagnosis consists of explaining why it is not. We represent the DSN as a collection of conceptual processes and search for a subset that explains the input-output discrepancies. This is done through a two-phase process. First, qualitative reasoning is used to formulate a plan consisting of a minimal sequence of processes that might explain discrepancies. The plan is then quantitatively executed (simulated). If the plan-execution fails to explain the discrepancy, a process of replanning ensues. Once the processes responsible for a discrepancy have been identified, the next step is to determine if those processes can be changed by parameter adjustment to remove the discrepancy. For this we use models that capture the causality relationships between the conceptual processes and actual system parameters.

C. EXPECTATION SOURCES

The diagnosis system design requires the availability of alternative sources of information (Expectations) that can be compared with the tracking output to detect discrepancies.

In the system development context, DSN developers can supply *a priori* knowledge of aircraft movements for controlled system tests. In this situation a diagnosis system could serve as an automated assistant to system developers. Although this is a potentially important role for a diagnosis system, it is equally important to consider diagnosis of a fully automated and operational DSN. This requires that discrepancy detection be fully automated. As described below, we have identified four strategies for providing expectations in a fully automated system. In all cases it should be noted that diagnosis includes the possibility of errors in expectations.

The most obvious strategy for discrepancy detection in the DSN context is to compare two different nodal views of an area of coverage which is common to both nodes. In this case, we consider one of the views to be the expectation and the other one as the candidate for diagnosis.

A second option is to provide tracking constraints by acoustic recognition processing of microphone signals. The basic idea is that the microphone signals contain information that is not

† R. Davis, "Diagnosis via Causal Reasoning: Paths of Interaction and the Locality Principle," in *Proceedings of AAAI-83* (August 1983), pp. 88-94.

utilized by the tracking system. Acoustic recognition processing can capture some of this information. A simple example of acoustic recognition processing is the detection of power peaks in the microphone signals at a node. A peak theoretically corresponds to an aircraft at its closest point of approach (CPA) with respect to the node. The time and power of this CPA-associated peak can be used to check the tracker output.

The third strategy is track extrapolation — predicting the evolution of currently measured tracks. The prediction may be based on knowledge about typical flight characteristics for various aircraft types and situations. The idea is to use the predictions for discrepancy detection in future track outputs.

The fourth strategy is to use empirical knowledge to identify unlikely track patterns. For example, tracks of short duration often do not correspond to actual targets. One might assume that long tracks are correct and that short tracks are discrepancies that need to be diagnosed.

Current plans are to develop elements of an experimental diagnosis system in the DSN development context, with expectations provided by knowledge of controlled test scenarios. We also plan to perform acoustic recognition experiments to develop track constraining algorithms for future use.

V. VIDEO SENSOR SUBSYSTEM

We are developing a video subsystem to demonstrate acoustic cueing, to improve tracking through the use of video-derived azimuth measurements, and to experiment with resource management and cooperative use of multiple sensor types.

The hardware for the video sensor subsystem (TV node) is complete except for planned modifications of the AZ/EL mount position readout electronics. The system includes an environmentalized, silicon target TV camera with a 16/160-mm zoom f1.8 lens, mounted on a remotely controlled AZ/EL mount, all mounted on a pedestal on the roof of a Lincoln Laboratory building. From this location we have a clear view of Hanscom Field and the surrounding area and of many targets of opportunity which can be used for system testing. All TV node electronics are installed in a cabinet rack inside the Laboratory. The system includes a standard computer terminal that is used for control.

Software has been written to test the basic functions of the TV system:

(a) Azimuth Positioning of Camera Mount (0° to 360°)

(b) Elevation Positioning of Mount (–20° to +30°)

(c) Camera Focal Length Setting (Zoom lens 16 to 160 mm)

(d) Read Position (Azimuth, Elevation and Focal Length)

(e) Relative Azimuth Positioning (Left or Right)

(f) Relative Elevation Positioning (Up or Down)

(g) Frame Freeze (acquire single frame)

(h) Mask (protect a portion of a frame)

Additional diagnostic and support routines have been written including routines to display images and make hard-copy printout, reset and reconfigure I/O display look-up tables, and print out frame buffer status.

Exploratory versions of image-processing routines have been implemented during the past quarter. One such routine, a moving-target detector, performs sequential TV frame subtractions and presents the results on the video monitor display. In tests with local aircraft, automobile, and pedestrian traffic, the display clearly shows the outline of the moving objects. By line-scanning such frames, frozen in a buffer, it should be possible to detect and determine the azimuth of a moving target. A second image-processing routine, an experimental 3×3 convolution program, demonstrates the potential uses of image averaging and/or smoothing as well as target edge detection techniques.

VI. ACOUSTIC SUBSYSTEM

A. ARRAY PROCESSING ALGORITHM EVALUATION

In this reporting period, we continued to evaluate the test-bed signal-processing algorithm with real and synthetic data.

First, we examined the sensitivity of the algorithm to changes in its parameter values using acoustic data from an F-111 jet aircraft. We also had ground truth information in the form of an independent radar track. Parameter variations that were investigated included the number of elevations sampled, averaging interval, and prefiltering frequency band. The algorithm was insensitive to the number of sampled elevations (ranging from 8 to 60 samples). Its performance improved with increasing averaging time interval, from 1 up to 10 s. This was expected since increased time averaging improves signal-to-noise ratio. For long-range targets, performance improved when prefiltering included the low-frequency bands.

F-111 runs also were used to investigate azimuth measurement precision. For this purpose power measurements were made every 0.25° in azimuth, and target azimuth was estimated as the azimuth corresponding to the largest power. Measured standard deviations of azimuth estimates ranged from about 0.6° for very close and high-power situations to about 4° at longer ranges near the detection limits of the system. These measurements are for low-wind good propagation conditions.

We plan to use 3° azimuth increments for our real-time implementation. This will be sufficient to get the best possible measurements for distant low signal-to-noise targets. Some accuracy may be lost under very good propagation conditions and for nearby targets, but we believe that this will not be a serious problem. The primary reasons for limiting the azimuth increment size are to conserve processing time and memory. Figure VI-1 shows the azimuth measurements obtained from a typical F-111 flyby using 3° increments. The plot shows azimuth corresponding to the largest power peak for each measurement time. This explains the very large errors early and late in the data. They correspond to times when the target is no longer the loudest noise source. The solid line is the predicted ground truth acoustic azimuth that was derived from independent radar tracks.

Existing 4-node UH-1 helicopter data also were reprocessed with the new signal-processing algorithm to provide target detections and azimuth measurements as test input for new tracking algorithms.

The new tracking algorithm will require an estimated standard deviation along with each azimuth estimate. We have begun to investigate how to provide this estimate and to investigate the performance differences between our present wideband signal-processing algorithm and the previous narrowband maximum likelihood method (MLM). Initial experiments are being performed with simulated data consisting of band-limited random plane waves and additive random noise. A signal band from 50 to 100 Hz was used. Wideband processing was performed using this entire

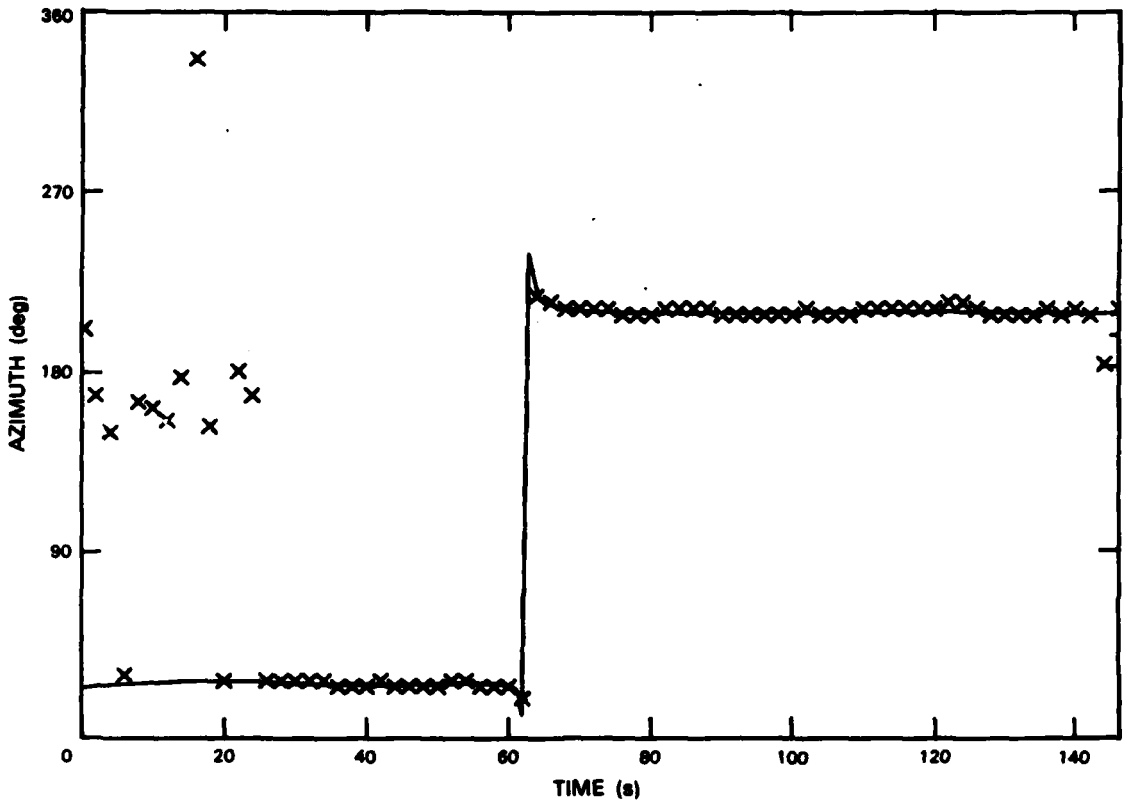


Figure VI-1. Acoustic azimuth measurements for F-111 flyby.

band, and MLM processing was performed from 48 to 104 Hz in 8-Hz steps and with 8-Hz resolution. The accuracy of the MLM depended upon the frequency of analysis. In general, the wideband method provided an accuracy corresponding to that for the MLM frequency with the best accuracy.

B. REAL-TIME SIGNAL-PROCESSING IMPLEMENTATION

The design for a new Signal-Processing System has been completed. An initial test version has been developed to run under the RSX11M operating system to test out the basic design. The test software reads prerecorded acoustic peak measurements from tape and provides them to the tracking system interface. The tracking system interface requires further refinement and we must also integrate existing array processor code into the system and develop an RSX11 driver for the A/D system. We are also investigating how to implement the same capabilities in an RSX11S operating system environment. RSX11S differs from RSX11M in that the former is a memory resident system and the latter requires a disk.

C. MICROPHONE ARRAYS

A new nine-element DSN acoustic array has been deployed into a forested area near the Laboratory. It is nearly equidistant from two other fixed arrays located on Laboratory building rooftops. We have located all electronic equipment associated with this array in our main DSN laboratory area, which is several hundred meters from the array. The new array will be used to test changes to front-end hardware and provide answers to questions concerning ground deployment of arrays in wooded areas. Standard unshielded telephone lines are used to carry the array signals. This required that we use differential amplifiers with high common-mode rejection at the terminating end.

Phone-line tests were conducted using Geosource Model MDS-10 preamplifier modules as replacements for existing front-end modules. The new units offer several advantages. They have selectable anti-alias filters with 750- and 375-Hz corner frequencies and their roll-offs are 117 dB/octave compared with a single corner at 250 Hz and a 12-dB/octave roll-off for the existing filters. They provide selectable gains from 12 to 48 dB in 4-dB steps, whereas the existing system provides only a single 37-dB gain setting. The newer amplifiers provide increased common mode rejection and lower noise. They offer selectable 60-Hz notch filters and high-pass filters to remove wind noise if necessary.

As a result of successful data-recording tests with the new array, using phone-line transmission and MDS-10 signal conditioning, 14 phone-line pairs have been allocated for use with the new array. A new prototype DSN front-end chassis has been constructed which is designed to accommodate MDS-10 modules and provide front panel switching of the various parameters for each individual channel. The decision to build new front ends for the remaining nodes will be made after we gain more experience with the new configuration.

VII. TEST-BED FACILITY ENHANCEMENTS

A. LL-XN HOST COMPUTER

During this reporting period, the Group host computer (a DEC PDP-11/70) was replaced by a DEC VAX-11/780 CPU. One new disk drive was added to three existing 256-Mbyte removable-media drives to provide a total formatted capacity of 1 Gbyte. The previously used magnetic-tape drive, array processor, printer, and network interface equipment were integrated into the upgraded environment. A newly purchased laser graphics printer was added to support higher resolution graphics hard-copy output.

An enhanced version of AT&T's VAX UNIX from the University of California at Berkeley (4.2BSD UNIX) was installed as the host operating system. An earlier version of the UNIX operating system, UNIX Version 7, was used on the previous Group host. The continued use of the UNIX operating system allowed for upward compatibility of much of the installed hardware and application software base, thus minimizing the effort necessary to accomplish a successful upgrade.

Since system hardware and software integration was completed, the system has resumed full user load and now functions as the primary host system for our research work. A significantly greater reliability and system response time have been realized with this upgrade, resulting in enhanced productivity on the part of system users.

B. NETWORKING

The installation of the VAX was done in parallel with the installation of an Ethernet to interconnect all Group computer resources, including DSN test-bed nodes.

The VAX system was easily integrated into the local network environment because of the generic network software support available in the Berkeley 4.2BSD UNIX operating system. The VAX and a separately procured Silicon Graphics were connected to our Ethernet and now communicate with each other. In addition, a SYMBOLICS 3600 Lisp Machine was connected, via the same physical Ethernet connection, to the host VAX with the aid of a special Chaosnet software module procured from SYMBOLICS for the VAX. An additional VAX-11/780 has also been connected to the DSN Ethernet and communicates to the ARPANET with the LL-XN host acting as an internet gateway processor.

To summarize, there are now two VAX-11/780s, a Silicon Graphics Workstation, and a Symbolics 3600 Lisp Machine connected to the network. In addition, a parallel effort (described below) has been under way to provide Ethernet communications between SNCs of the actual DSN test-bed. Presently, four of the test-bed nodes are attached to the DSN Ethernet.

The DSN Ethernet will provide improved throughput and experimental flexibility for experiments involving multiple test-bed nodes. The Ethernet replaces a serial line-based communication

emulation system. Internodal Ethernet communications software for the SNCs has been developed and installed. The software provides an internodal datagram service, with both broadcast and point-to-point messages. To ease the transition from the old serial line-based system to the new Ethernet system, the applications program interface to the communications software has remained unchanged.

C. RADIO COMMUNICATION

Implementation of the radio broadcast protocol for the Communication Networks Technology radios has proceeded along two lines. The lower layer of the protocol has been implemented in an SNC processor connected with a radio unit interface board in a test configuration. Additionally, the higher layer of the protocol has been implemented in a UNIX simulation.

Several portions of the lower layer software have been finished. DMA software has been completed for the Intel 8089 input/output processor which is a component of the Radio Unit Interface (RUI) board. In addition, we have developed the low-level software interface between the Digital Control Unit (DCU) and the RUI which enables programs in the DCU to control the radio.

The higher layer of the protocol utilizes the functions provided by the software interface to the RUI. The functions of the higher layer include scheduling of packets, rescheduling of aborted transmissions, data transmission, packet reception, and delivery of received packets to user processes.

We are now completing the lower layer of the protocol and integrating the higher layers into the nodal run-time system. Development work for the radio protocol is being carried out in two SNC chassis equipped with RUI boards. For testing purposes, the boards may be configured in loopback or in a back-to-back connection.

D. NODE FLOATING POINT

Floating-point arithmetic hardware and software have now been added to the SNCs. SKY-FFP floating-point processors have been installed and are operational on all standard nodal computers. Needed changes have been made to the C cross-compiler to support both double- and single-precision floating-point arithmetic. Additional routines have been added to the nodal run-time system to permit one FFP to be shared by all processors in the SNC.

A UNIX-compatible floating-point library was developed for use in the SNC. The library includes a number of trigonometric and arithmetic functions for both single and double precision. In addition, the UNIX math library was extended to provide single-precision arithmetic.

GLOSSARY

CPA	Closest Point of Approach
CPU	Central Processing Unit
DCU	Digital Control Unit
DEC	Digital Equipment Corporation
DSN	Distributed Sensor Networks
MACSYMA	A symbolic mathematics manipulation program
MLM	Maximum Likelihood Method
NRTS	Nodal Run-Time System
RUI	Radio Unit Interface
SATS	Semiannual Technical Summary
SNC	Standard Nodal Computer
SPS	Sound Processing Subsystem
UIC	User Interface Computer
UIP	User Interface Program

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ESD-TR-84-326	2. GOVT ACCESSION NO. AD-A158033	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Distributed Sensor Networks		5. TYPE OF REPORT & PERIOD COVERED Semiannual Technical Summary 1 April — 30 September 1984
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR*(s)* Richard T. Lacoss		8. CONTRACT OR GRANT NUMBER*(s)* F19628-85-C-0002
9. PERFORMING ORGANIZATION NAME AND ADDRESS Lincoln Laboratory, M.I.T. P.O. Box 73 Lexington, MA 02173-0073		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Program Element Nos. 62708E Project Nos. 5D30 & 5T10 ARPA Order 3345
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209		12. REPORT DATE 30 September 1984 13. NUMBER OF PAGES 40
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)* Electronic Systems Division Hanscom AFB, MA 01731		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

multiple-sensor surveillance system
distributed tracking
target surveillance and tracking
communication network
acoustic sensors
video sensors
low-flying aircraft
acoustic array processing
digital radio
distributed estimation
knowledge-based data interpretation

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This report describes the work performed on the DARPA Distributed Sensor Networks Program at Lincoln Laboratory during the period 1 April through 30 September 1984.

Keywords include:

DD FORM 1 Jan 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*